

REMARKS

The present Response is submitted in reply to the Office Action of June 14, 2004 in the continued examination of the above identified Application.

Claims 1-16 are presently pending in the Application and the Examiner has objected to claim 11 for an informality therein, has rejected claims 1, 3, 5, 9, 11, 13 and 15 under 35 U.S.C. § 102 over U.S. Patent No. 6,205,449 to Rastogi et al. '449 for a SYSTEM AND METHOD FOR PROVIDING HOT SPARE REDUNDANCY AND RECOVERY FOR A VERY LARGE DATABASE MANAGEMENT SYSTEM, hereafter referred to as "Rastogi et al. '449", and has rejected claims 2, 4, 6, 8, 10, 12, 14 and 16 under 35 U.S.C. § 103(a) over Rastogi et al. '449 and further in view of U.S. Patent No. 5,513,314 to Kandasamy et al. for a FAULT TOLERANT NFS SERVER SYSTEM AND MIRRORING PROTOCOL, hereafter referred to as "Kandasamy et al. '314".

The Applicant acknowledges and respectfully traverses the raised obviousness rejection in view of the following remarks.

The following will first review previously presented discussions of both the present invention and the teachings of Rastogi et al. '449 and the distinctions of the present invention as claimed over the teachings of Rastogi et al. '449, and will then similarly discuss Kandasamy et al. '314 and Rastogi et al. '449 in view of Kandasamy et al. '314. It will be noted that the Applicant has restated certain of the discussions and arguments previously presented in an attempt to clarify both the present invention and the distinctions of the present invention over the cited prior art.

First considering the present invention, the present invention and the claims of the present application are directed to system resource for performing system resource operations requested by a client of the system resource, such as a file server performing file transactions requested by a client. According to the present invention and the claims, the system resource includes a state machine logging mechanism for extracting, storing and restoring state machine information representing the state of operation of the system resource and of the system resource operations, such as file transactions.

The claims recite and define a system resource, such as a file server, with a state machine logging mechanism as including a system resource sub-system, such as a storage sub-system, and a control/processing sub-system that includes a resource control processor performing system resource operations in response to client requests and controlling operations of the system resource sub-system. The control/processing sub-system may be, for example, a file system processor performing file transaction operations in response to client requests and controlling file storage operations of the storage sub-system, and the state machine logging mechanism.

The state machine logging mechanism, in turn, includes a state machine log generator for extracting state machine information defining a current state machine representing a current state of execution of a system resource operation and a state machine log mechanism for storing a sequence of one or more state machines representing the sequential, detailed, state machine level operations of the system in executing the requested transactions. The state machine log generator is responsive to the restoration of operation of the system resource after a failure of the system resource for reading the state machine information from the state machine log mechanism and restoring the state of execution of a current system resource operation.

As described in the specification and as recited in the claims, the state machine log mechanism includes a state machine log mirroring mechanism operating separately from the control/processing sub-system and communicating with the state machine log generator through a local high speed data link for receiving and storing mirror copies of the state machine information. The state machine log mirroring mechanism may then respond to a failure of the control/processing sub-system and to a subsequent restoration of operation of the file server after a failure of file server operations to read the state machine information from the state machine log mirroring mechanism and to restore the state of execution of a file transaction.

In this regard, the state machine log mirroring mechanism operates separately from but concurrently with and in cooperation with the control/processing sub-system in such a manner that the state machine log mirroring mechanism is capable of remaining in operation even upon failure of the control/processing sub-system. That is, the control/processing sub-system and

state machine log mirroring mechanism operate concurrently and cooperatively but are independent from one another in such a manner, for example, by being powered from separate power sources, that a failure in the control/processing sub-system will not in itself result in a failure of the state machine log mirroring mechanism.

As will be apparent from the following discussions, and from previous discussions of the present invention and the cited prior art, a comprehension and understanding of "state machines", "machine state", and "state machine information" is necessary to a comprehension and understanding of the fundamental distinctions of the present invention over the cited prior art.

As previously discussed and as described, as defined in the specification and in the claims and as well known in the relevant arts, a "state machine" is a representation or model of a machine or system, such as a system resource, a file server, a sub-system or logical or functional element of a system or sub-system of any form, that executes operations as a sequence of discrete "machine states", also referred to as "states". Each "machine state", or "state" of a sequence of machine states is defined during or at an interval or point in time during in the machine state is not changing, which is typically and, for example, during the interval between clock pulses. The present state and next operating states of a state machine are described and defined by and are comprised of the current state of the machine, the state functions of the machine itself, that is, the logic and circuit functions implemented in the machine that determine the responses or changes in state of the machine as a result of a current operating state and any current control or data inputs that effect the current state, and any control or data inputs that will effect the present or next state of the state machine.

In summary, therefore, a "state machine" is therefore comprised of and is defined and described as a sequence of one or more state machines wherein each state machine in the sequence of state machines is defined by the current state at that point in time. That is, each state machine in a sequence of one or more state machines is defined and described by the control and data values residing in the machine, and the state functions of the machine, that is, the functions or operations that will be executed by the state machine to result in the next state

machine, and any current control or data inputs that will operate or be used in defining the next state machine in the sequence.

The state functions of a system, that is, the responses or changes in state of the machine as a result of a current operating state and any current control or data inputs that effect the current or next states of the system, are determined by the logic and circuit functions implemented in the system determine. As such, the state functions of a given system are typically fixed and are thereby explicitly or implicitly known and need not be specified for each state machine individually. As such, the state machine of a system at any point in the operations performed by the system is fully defined and described by the machine state at that point, that is, by the control and data values residing in the state machine or present as inputs to the state machine at that point. A sequence of operations executed by the system may, therefore, be defined and described by a corresponding sequence of machine states.

At this point, it must be emphasized that the state machine logging mechanism of the present invention extracts and stores "state machine information" defining one or more sequential "state machines" during the execution a resource operation or transaction rather than storing information pertaining to the system operation only at the start or conclusion of an operation or transaction, as is typical in transaction logging type systems.

In contrast, and by way of illustration, conventional systems of the prior art such as described in Rastogi et al. '449 and Kandasamy et al. '314 typically record the operations of a system by recording the parameters of an operation to be performed at the start of the operation and possibly at the end of the operation and the recorded "parameters" typically include only the command, request or instruction initiating the operation and the input data provided to the operation or the output data resulting from the operation. A conventional logging operation for a system operation may thereby be thought of as a system capturing a "snap-shot" of the minimum information required to define an operation, that is, the instruction, request or command initiating the operation and the data operated upon in the operation. For example, at column 8, lines 3-15, Rastogi et al. '449 defines a "transaction" as being comprised of a sequence of operations at some level in the system and states that the Rastogi et al. '449 system records "transactions". As such, and by Rastogi et al. '449's own

statement, the Rastogi et al. '449 system does not record or restore the intermediate sub-operations within an "transaction".

As such, a conventional logging system of the prior art, such as Rastogi et al. '449, is of "low granularity", operating only on a per operation basis, and does not and cannot record or restore the intermediate sub-operations within an operation. Instead, and for example, a conventional logging system can record and restore only completed, entire transactions.

It is well understood by those of ordinary skill in the relevant arts, however and, in fact, is described in Rastogi et al. '449', that any operation in a system, such as a file transfer, a file read or a file write, is comprised of a sequence of sub-operations wherein, as recognized in the state machine mechanisms of the present invention, each sub-operation is a complete operation in itself and essentially involves a single operation performed on a single body of data. A given operation or a completed portion of an interrupted or aborted operation can, therefore, be accurately and completely restored from a log record only by accurately and completely recording the sub-operations of the sequence of sub-operations comprising the operation, which is in accordance with the prior art limitation that only fully completed operations are logged and can be restored.

The present invention, however, provides a mechanism for logging and mirroring each of the sub-operations of the sequence of sub-operations comprising a given operation of the system by recording each sub-operation as a state machine, that is, as the machine state of a state machine, of a sequence of state machines representing or modeling the operation of the system.

Stated simply, therefore, the state machine logging mechanism of the present invention thereby provides greater "granularity" in recording the execution of an operation or transaction in capturing and storing of a sequence of state machines reflecting the sequential sub-operations of the system during the execution of a requested transaction, thereby recording and representing each transaction or operation as a sequence of state machines. Because a state logging and mirroring mechanism of the present invention will record each of the system sub-operations that comprised a system operation or transaction as an individual state machine of a sequence of state machines, a state logging and mirroring mechanism of the present

invention can record events within the execution of any system operation or transaction. This greater granularity in turn allows the restoration of transactions and operations at the point at which they were interrupted, that is, during the execution of any of the sub-operations comprising a transaction or operation, rather than at only a single point at the beginning or end of the transaction or operation.

Stated again for emphasis, the state machine logging mechanism of the present invention extracts and saves such state machine information at each sub-operation or state, during the performance or execution of a resource operation or transaction, so that each transaction or operation is represented and recorded as a sequence of state machines wherein each state machine represents a single sub-operation within the transaction or operation. Also, and again, each state machine is comprised of the state machine information comprising and defining a state machine at at each point in the performance or execution of the operation or transaction.

The state machine logging mechanism of the present invention thereby allows the restoration of a resource server or transaction at a point during the performance or execution of the operation or transaction. The state machine logging mechanism of the present invention thereby also provides greater assurance that a given resource server operation or transaction is captured for subsequent restoration, if needed, because the necessary information is captured at each stage in the execution of the transaction as a sequence of points in the execution of the operation or transaction, rather than only when the operation or transaction is completed. Stated another way, each change in the system during the execution of a transaction defines a new state of the machine, which is captured and stored as the next state machine in the sequence of state machines defining the transaction. As a consequence, no essential step during the execution of a transaction is or can be lost, thereby allowing reliable reconstruction or restoration of any transaction.

Lastly, it must be noted that the invention as described in the specification and as recited in certain of the claims is directed to an implementation of the present invention in a system resource having a single control/processing sub-system having an associated state

machine log generator and a state machine log and is recited as such in claims 1, 3, 5, 7, 9, 11, 13 and 15.

The present invention as recited in the claims is, however, also directed to an implementation of the present invention wherein the state machine logging mechanism further includes a state machine log mirroring mechanism that is operationally separate and independent from the control/processing sub-system, such as in claims 2, 4, 6, 8, 10, 12 14 and 16. That is, in the recitations of claims 2, 4, 6,8, 10, 12, 14 and 16, the state machine logging mechanism has a state machine log situated in or in association with the control/processing sub-system and a separate state machine log mirroring mechanism that is effectively external and independent of the control/processing sub-system. In this regard, it must be recognized that the state machine log mirroring mechanism of a given control/processing sub-system is not independent from the control/processing sub-system or the state machine log of that control/processing sub-system as regards the state machine log mirroring functions, but is separate and independent from that control/processing sub-system as regards basic support functions, such as power to the separate state machine log mirroring mechanism. As such, and for this reason, the separate state machine log mirroring mechanism can continue operation if there is a failure of the control/processing sub-system.

Further in this regard, it must be noted that the state machine log mirroring mechanism is not a separate system analogous to the control/processing sub-system and is not a complete, self-contained and full function system in the manner of a general purpose computer. Instead, the state machine log mirroring mechanism is a specific purpose mechanism designed and constructed only to perform the state machine log mirroring function of capturing and storing a copy of the state machine information and state machines that are stored in the state machine log associate with the state machine log generator that is associated directly with the control/processor sub-system. As such, it will be clear to those of ordinary skill in the arts that the separate state machine log mirroring mechanism does not and cannot operate as a "back-up" unit for the control/processing sub-system in processing requests for operations as the state machine log mirroring mechanism simply does not have the functionality for this type of operation. The state machine log mirroring mechanism instead operates solely as a specific

purpose mechanism designed and constructed only to perform the state machine log mirroring function of capturing and storing a copy of the state machines comprising the control/processing sub-system while it is processing transactions.

It must also be noted that the claims are also directed to implementations of the present invention having dual control/processing sub-systems with associated state machine logging mechanisms and state machine log mirroring mechanisms. In these implementations, each control/processor sub-system has an associated state machine logging mechanism for its own transactions or operations and a state machine log mirroring mechanism associated with the other control/processing sub-system. As such, each control/processing sub-system maintains its own state machine logging mechanism and maintains a state machine log mirroring mechanism for the other control/processing sub-system and processes transactions independently of the other control/processing sub-system.

In this regard, it must also be noted that in the system resources or file servers having dual control/processing sub-systems, the two control/processing sub-systems operate concurrently, independently and in parallel with each other and with each control/processing sub-system performing its own system resource operations or transactions, and with each control/processing sub-system providing only a residence and support for the state machine log mirroring mechanism of the other control/processing sub-system.

As such, and as described in the specification and recited in the claims, neither control/processing sub-system provides "back-up" for the other control/processing system in the sense of standing by ready to perform or execute transactions or operations for the other control/processing sub-system if the other control/processing sub-system should fail. It must also be noted that because of the concurrent, independent and parallel operation of each control/processing sub-system, with each control/processing sub-system executing only the requests for transactions or operations that are directed to it.

In this regard, it must be noted that in the systems of the present invention, and in fundamental contrast from the typical form of "back-up" systems as taught by, for example, Rastogi et al. '449, the full processing power of the two control/processing sub-systems is available at all times to process requests from clients. If one control/processing sub-system

fails, the other control/processing sub-system continues operative to perform the request directed to it and to maintain the state machine information necessary to subsequently restore the failed control-processing sub-system. In addition, the operative one of the control/processing sub-systems may perform requests that would be directed to the other control/processing sub-system if those requests are re-directed to the operative control/processing sub-system. It will, therefore, be apparent to those of ordinary skill in the arts that only in the event of a failure in one of the control/processing sub-systems of the present invention is the total transaction processing capability of the system reduced to that levels provided by the normal operation of the "back-up" systems of the prior art, such as taught by Rastogi et al. '449.

Further in this regard, and in fundamental contrast from the system of the present invention, in conventional "back-up" systems, such as that taught by Rastogi et al. '449, there are typically two identical systems arranged in parallel with one unit being designated as the primary unit and the other as the "back-up" unit. The primary unit typically performs all operations requested of the system, while the "back-up" unit is typically idle until it is required to replace the other unit in performing the operations, so that only one half of the total power and resources of the system are available at any time. The system of the present invention, however, makes significantly more of the potential control/processing sub-system capability available to clients of the system because both halves of the system are fully operatable at all times unless and until one of the control/processing systems fails. In complete contrast, in a conventional system such as taught by Rastogi et al. '449, only one half of the potential processing power of the system is available at any time, even when neither of the systems has failed.

Next considering the teachings of Rastogi et al. '449. Rastogi et al. '449 describes a system having "hot spare" support wherein the system includes two identical computer systems communicating through a network and wherein, at any given time, one of the computer systems is designated as the "primary" computer system and the other is designated as the "secondary" computer system. The computer system that is designated as the primary computer system performs all operations, that is, all transactions, of the system and generates a log of all such

transactions. The computer system that is designated as the secondary computer system does not perform any operations while the primary system is in operation, but is instead a "back-up" system that stores a copy of the transaction log of the primary system. If there is a failure in the current primary computer system, the secondary computer system becomes the primary computer system and assumes execution of all transactions directed to the system, starting with the copy of the primary system transaction log stored in the secondary computer system, which then becomes the primary computer system.

The primary and essentially the sole object of storing a copy of the current primary system transaction log in the secondary system is to maintain the primary and secondary systems in "synchronization" so that the secondary computer system can assume execution of the transactions directed to the system with minimum loss of the preceding transactions that have been executed by the computer system that was previously the primary computer system.

Rastogi et al. '449 states that, for this purpose, the primary and secondary computer systems each stores a record of the "state" of the computer systems. It will be readily recognized by those of ordinary skill in the relevant arts, however, that Rastogi et al. '449's use of the term "state" differs fundamentally from the term "state" as used in the specification and claims of the present Application.

In particular, in Rastogi et al. '449 the term "state" refers to bits of information stored in the primary and secondary systems that indicate, in each system, whether the system is the current primary system or is the current secondary system and whether the two computer systems are in "synchronization", that is, have matching copies of the primary computer system transaction log. At any given time, only one computer system can be the current primary computer system and can execute the system transactions. The secondary system will always be idle as regards the execution of transactions and will function solely to store a copy of the primary computer system transaction log while waiting to assume execution of the transactions upon failure of the current primary computer system.

It will, therefore, be apparent to those of ordinary skill in the arts that in the teachings of Rastogi et al. '449 the term "state" has no relationship or meaning with regard to the state of operation of a system at each step in executing a transaction, or even to the actual execution

of transactions, but instead relates only to the overall current functional assignments of the two systems and, in particular, to which one is assigned to execute transactions and which one is assigned to store a copy of the transaction log, and not how the transactions are executed.

As taught by Rastogi et al. '449, the primary system transmits a copy of the primary system transaction records to the secondary system through a network connection in either of two circumstances, depending upon the specific designed operation of the Rastogi et al. '449 system. In one circumstance, the primary system transaction log is transmitted to the secondary each time the transaction log in the primary system is "flushed" to disk, that is, is moved from the primary system memory space to the primary system mass storage device for long term storage. In the second circumstance, the primary system will flush its local copy of the transaction log to its own disk when the primary system has transmitted a copy of the transaction log to the secondary system and has received an acknowledgment of the transmission from the secondary system.

It is, therefore, apparent that the present invention is distinguished over and from the Rastogi et al. '449 system for a number of fundamental reasons, which are recited in the claims as amended herein.

For example, and in complete and fundamental contrast from Rastogi et al. '449, the system of the present invention captures and stores state machines representing the detailed operation of the system in the execution of each transaction and stores these state machines while the corresponding transaction is being executed, so that the execution of a transaction may be continued or resumed at any time, not just at the end or beginning of a transaction.

It will, therefore, be apparent from the above discussions that the present invention is fundamentally distinguished over and from the system taught by Rastogi et al. '449 in essentially every respect. For example, the transaction log of the Rastogi et al. '449 system captures or generates transaction records only when the transactions are completed, that is, at the end of the execution of each transaction.

In fundamental contrast from the present invention, which captures each transaction and the sub-operations within each transaction as a sequence of state machines, Rastogi et al. '449

does not and cannot capture and preserve records of the sub-operations within and comprising a transaction or operation.

This distinction is further supported and emphasized in that the Rastogi et al. '449 system actually records or stores a record of a completed transaction only at the conclusion of the transaction, when the record is transferred into mass storage, as any record that the Rastogi et al. '449 system generates before the completion of a transaction is held only in volatile memory and is thus lost if the system fails. In a like manner, a copy of a transaction record is transmitted to and stored in the secondary computer system only when the record is transferred to the primary system mass storage, so that the back-up copy of the transaction record can also be lost if a system failure occurs before the end of the transaction.

In fundamental contrast from Rastogi et al. '449, the system of the present invention captures and records and mirrors each sub-operation in an operation as a state machine and as and when each sub-operation is executed, so that a transaction or operation is continuously recorded at each sub-operation point as the transaction or operation is executed. As a consequence, if there is a failure of a control/processor sub-system during the execution of a transaction, the sub-operations of the transaction will have been captured and recorded and mirrored up to the sub-operation in which the failure actually occurred. As such, only a part of a transaction is lost and the transaction can be restored up to the point of failure. The Rastogi et al. '449 system cannot and does not provide this level of capture and mirroring and can capture and restore only completed transactions.

In further fundamental contrast from the present invention, it must be noted that the transaction records captured by Rastogi et al. '449, that is, the data comprising a record of a completed transaction that is captured by Rastogi et al. '449, has no relationship to a state machine representing the control and data values present in a state machine system and representing the operating state of a system of operation at a given time.

To explain in greater detail, it must first be noted that Rastogi et al. '449 describes a transaction as a sequence of operations at one or more levels and describes a transaction log as storing such transactions. Rastogi et al. '449, however and, for example, at column 8,

lines 3-15, speaks only of recording completed transactions and not of capturing, recording or restoring the sub-operations of a transactions.

Further in this regard, it must be noted that the only mention of "state" by Rastogi et al. '449 is, for example, at column 5, lines 29, to column 7, line 15, wherein Rastogi et al. '449 defines "state" within the context and meaning of the teachings of Rastogi et al. '449 as being no more than stored information indicating which is the primary system and which is the secondary system and whether the primary and secondary systems are synchronized.

Thus, while Rastogi et al. '449 uses the word "state", it must be noted and understood that Rastogi et al. '449 defines the term "state" to have a meaning within the teachings of Rastogi et al. '449 that has no relationship to the defined meaning of the term "state" in the present Application and the claims thereof. In this regard, it must also be noted that both usages of the word "state" are correct as being within the commonly accepted meanings of the word, but that the uses have separate and different meanings that are dependent upon the contexts in which they are used. Therefore, while Rastogi et al. '449 uses the word "state", the meaning of the word "state" as employed and defined in the present Application and claims is fully and fundamentally distinguished from the meaning of the word "state" as used in Rastogi et al. '449.

Further in this regard, it must be noted that Rastogi et al. '449 essentially only describes representing a transaction in the primary system transaction log in terms of what are, in fact, the instructions or commands initiating each operation at each level, and perhaps any data input to the operation. At no point does Rastogi et al. '449 describe or even mention a "state machine system", a "state machine", system "state" as represented by the control and data values residing in the system at a given point during the execution of a transaction and defining the "state machine" of the system at that point.

The only reasonable conclusion that could be reached by one of ordinary skill in the arts is, therefore, that Rastogi et al. '449 is, as stated, referring to a transaction log as containing only either the instructions or commands initiating each a transaction, and any input data, or the data resulting from a transaction at the completion of the transaction and that,

in complete contrast from the present invention as described and claims, Rastogi et al. '449 does not mention, describe or teach in any way the use of state machines to capture, record and mirror the sub-operations of transactions as state machines.

To further illustrate this distinction by an example, it must be noted that each sub-operation or step in the execution of a transaction is often effected by the outcome of a preceding sub-operation or step in the execution of the transaction and that, in most systems, any such information from a preceding sub-operation or step will be appropriately stored in the processing unit. A conventional system such as that taught by Rastogi et al. '449, however, may well miss such information from a preceding sub-operation or step within an operation as the log stores only the current instruction or command and input data for the operation as a whole or the data resulting from completion of the operation as a whole, and does not store data or information for the sub-operations or steps within the operation.

A state machine system of the present invention, however, because the state machine log stores, at each sub-operation or step in the execution of a transaction, a state machine representing the state of the system during that sub-operation or step. As described and recited in the present Application and claims, each such state machine includes the control and data values present in the system at that time and effecting the execution of the step, including information from a preceding sub-operation or step that may effect the execution of the current sub-operation or step.

The above discussed characteristics of the Rastogi et al. '449 system, that is, the logging of transactions rather than state machines and the logging of a transaction only when the primary system flushes its transaction log to local mass storage, results in yet other fundamental distinctions between the present invention and the teachings of Rastogi et al. '449.

For example, because the Rastogi et al. '449 system captures transactions, that is, the instructions and data initiating a transaction or at the completion of a transaction, rather than a sequence of state machines, the Rastogi et al. '449 system is essentially limited to capturing, recording and restoring each transaction as an entity. The Rastogi et al. '449 system, therefore, cannot and does not capture or record the sub-operations within a transaction and thus cannot restore a transaction any sub-operation within the execution of the

transaction, but instead captures only the beginning or end of a transaction. It is for these reasons that the Rastogi et al. '449 system contains both a "redo" log, so that a transaction can be re-executed from the start, and an "undo" log, so that a transaction can be "undone", or canceled, by "undoing" the transaction from the end.

These fundamental distinctions between the present invention and the teachings of Rastogi et al. '449, are further supported by the fact that the Rastogi et al. '449 system "logs" a transaction in the back-up log in the secondary systems only when the primary system flushes its transaction log to local mass storage. In other words, the "log" of a given transaction is stored solely in the volatile memory of the primary system until the log is flushed to the primary system mass storage, and is not until the log is flushed to the primary system mass storage that the log is also "flushed" to the secondary system to be stored in the secondary system mass storage.

While Rastogi et al. '449 does not address the implications of these log or flush operations explicitly, it is clear that a "flush" of a transaction from memory to mass storage will typically and normally take place only at the conclusion of the transaction as this will thereby require the storage of only the minimum amount of information about the transaction in order to allow the transaction to be "redone" or "undone". This interpretation of the teachings of Rastogi et al. '449 are supported by Rastogi et al. '449 at, for example, column 8, lines 3-61, and in more detail at column 8, line 3 through column 11, line 8. It is noted, in this regard, that Rastogi et al. '449 does not in fact state that a log entry contains information taken during the execution of the transaction, but only that the stored information includes only the information essential to reconstruction of the transaction as an entity. The system description provided in Rastogi et al. '449, however, is compatible with a system that stores only the data and commands initiating a transaction or the data resulting at the completion of a transaction is completed.

It must also be noted in this regard that Rastogi et al. '449 explicitly teaches only the storage of transactions, such as at column 8, lines 3-61, for example, and does not discuss or even mention or suggest storing a sequence of state machines representing a transaction or any other form of sub-operation data from within the execution of a transaction.

Again, Rastogi et al. '449 speaks only of "redoing" and "undoing" a "transaction" as an entity, which is defined as a set of sub-operations and which requires only either the transaction starting data and commands or the transaction results. Rastogi et al. '449 does not even mention restoring or resuming a transaction at any of the sub-operations comprising the transaction, but only of redoing or undoing a transaction in its entirety.

As such, it is clear that Rastogi et al. '449 does not and cannot capture or store state information of any sort, including state machines, state machine data or the data and control values residing in the control/processing sub-system state machine at each step in the execution of a transaction. It is also clear to those of ordinary skill in the art that this distinction is so fundamental and so basic to the entire design and operation of the two systems as to render the teachings of Rastogi et al. '449 rerelevant with respect to the present invention.

Lastly in this regard, it is noted that the Examiner states that Rastogi et al. '449 teaches the use of state information because the secondary system is available "immediately" to take over the functions of the primary system upon failure of the primary system. It must be noted, however, that not only does Rastogi et al. '449 not even mention state, except with regard to which processor is the primary and secondary processor and whether they are synchronized, that the use of the term "immediately" as used in Rastogi et al. '449 must be read and interpreted in light of the remainder of the teachings in Rastogi et al. '449.

That is, and for example, Rastogi et al. '449 clearly teaches that the logging of a transaction in the back-up log in the secondary systems occurs only when the primary system flushes its transaction log to local mass storage, which is typically only at the end of the transaction. This thereby indirectly states that not only will an "unlogged" or "unflushed" transaction be lost if there is a power failure to the primary processor before the log is flushed, but that since the logging takes place only at a "transaction rate", the term "immediately" does not imply any great speed of response but only that it will take place at the normal paces of transaction processing. This position is further supported when it is noted that the primary system and the secondary system in which the transactions are logged are interconnected through a network so that any restoration of transaction data must take place at network transaction speeds, which is normally a rather slow "immediately". A more reasonable

interpretation of the Rastogi et al. '449 teachings is that the secondary system is ready to assume the transaction processing operations of the primary system immediately upon being informed of a failure of the primary system, which is a very different matter and operation from the transaction logging operations.

Support for this position may be found in Rastogi et al. '449 at, for example, column 3, line 22, through column 4, line 59; column 5, lines 9 to 44; column 7, line 38 to column 8, line 2; column 8, lines 3-15, and column 8, line 24 through column 10, line 30.

To consider this distinction further, and in contrast from the Rastogi et al. '449 system, the log mechanism of the present invention is local to the transaction control/processing sub-system for communications purposes and, the log mirroring mechanism communicates with the log mechanism through a local, high speed datalink dedicated to that purpose. It is, therefore, in fact, the system of the present invention, and not the system of Rastogi et al. '449, in which logged transaction information, that is, transaction state machines, can be recovered "immediately" from the log mechanism or log mirroring mechanism. It is also only in the system of the present invention, and not the system of Rastogi et al. '449, that can gain benefit of the "immediate" restoration of a transaction record from the log mechanism or log mirroring mechanism because it is only the system of the present that stores "state machines" that enable the restoration of a transaction at any point during the transaction.

It is, therefore, clear and the belief and position of the Applicant, that the present invention is completely and fundamentally distinguished over and from the teachings of Rastogi et al. '449 under the requirements and provisions of 35 U.S.C. § 102 and 35 U.S.C. § 103 because Rastogi et al. '449 teaches and suggests only the capture and logging of transaction records representing the basic transaction operation commands, instructions and data and because Rastogi et al. '449 does not reach or even suggest in any way the capture, logging and restoration of transactions as sequences of state machines including the control and data values residing in and controlling the system at each step of the execution of a transaction.

It is further noted that the Examiner has, in fact, stated that Rastogi et al. '449 does not teach the capture, logging and restoration of transactions as sequences of state machines

including the control and data values residing in and controlling the system at each step of the execution of a transaction.

It is thus the belief and position of the Applicant that this distinction in itself is so fundamental and basic as to comprise a complete distinction of the present invention over and from the teachings and suggestions of Rastogi et al. '449 under the requirements of 35 U.S.C. § 102 and 35 U.S.C. § 103.

Finally, and in further distinction between the present invention and the teachings of Rastogi et al. '449, it must be noted that the primary and secondary computer systems of the Rastogi et al. '449 system do not correspond either structurally or functionally with the dual control/processing sub-systems of the present invention. That is, and as described, for example, in Rastogi et al. '449 at column 3, line 8 through column 4, line 2, the two computer systems of the Rastogi et al. '449 system are not parallel, cooperating sub-systems within a system, but are completely and separate computer systems.

In still further fundamental distinction between the present invention and Rastogi et al. '449, it must be noted that the dual control/processing sub-systems of the present invention operate independently but concurrently with each control/processing sub-system executing only the requests for transactions or operations that are directed to it, so that the full processing power of the two control/processing sub-systems is typically available at all times to process requests from clients. In basic contrast from the present invention, the two computer systems of the Rastogi et al. '449 system do not operate concurrently at any time. Instead, the primary processor alone operates to execute all transactions of the system while the secondary processor is always idle with respect to the execution of transactions, so that only one half the potential processing power of the Rastogi et al. '449 system is available at any time.

In addition, the primary and secondary computer systems of the Rastogi et al. '449 system do not correspond either structurally or functionally with the state machine logging mechanism and state machine log mirroring mechanism of the present invention as each of the primary and secondary computer systems of the Rastogi et al. '449 system is a full function, general purpose computer capable of performing both transaction operations and transaction logging. In contrast, the state machine logging mechanism and state machine log mirroring

mechanism of the present invention are both dedicated purpose, specialized function mechanisms that are structurally and functionally different from one another and are directed to separate and distinctly different functions. In a like manner, the control/processing sub-system and a corresponding state machine log mirroring mechanism are structurally and functionally distinguished from the primary and secondary computer systems of the Rastogi et al. '449 system for the same reasons.

It must also be noted that a control/processing sub-system and its associated state machine logging mechanism with the associated state machine log mirroring mechanism cannot be compared, structurally or functionally, with the primary and secondary computer systems of the Rastogi et al. '449 system because the primary and secondary computer systems of the Rastogi et al. '449 system are, in fact, identical but completely separate and independent systems from one another. In contrast, the state machine log mirroring mechanism is functionally an integral element of the corresponding state machine logging mechanism, even though the state machine log mirroring system resides separately from the state machine logging mechanism as regards such support functions as the power supplies so as not be involved in a failure of the corresponding control/processing sub-system with which the state machine log generator and log reside.

It is the belief and position of the Applicant that for the reasons discussed above the teachings of Rastogi et al. '449 do not and cannot describe or suggest the present invention as recited in the claims under either or both of 35 U.S.C. § 102 or 35 U.S.C. § 103. As such, the Applicant respectfully requests that the Examiner reconsider and withdraw the rejections of claims 1, 3, 5, 9, 11, 13 and 15 under 35 U.S.C. § 102 over Rastogi et al. '449, and the allowance of claims 1, 3, 5, 9, 11, 13 and 15.

Lastly with regard to Rastogi et al. '449, the Examiner has requested that the Applicant point out specifically the portions of Rastogi et al. '449's teachings that support the characterizations of Rastogi et al. '449 that have been described and discussed herein above. In reply, the Examiner will note that the Applicant has referred specifically to relevant portions of the teachings of Rastogi et al. '449 in the discussions herein above. For the convenience of the Examiner, however, the Applicant can summarize the references to Rastogi et al. '449

as including column 8, lines 3-15; column 5, line 29 to column 7, line 15; column 8, lines 3-61; column 8, line 3 to column 11, line 8; column 3, line 22 to column 4, line 59; column 5, lines 9-44; column 7, line 38 to column 8, line 2; column 8, line 24 to column 10, line 30, and column 3, line 8 to column 4, line 2.

Next considering Kandasamy et al. '314, the Examiner cites Kandasamy et al. '314 as teaching concurrent backup and mirroring of a file server by another file server.

Kandasamy et al. '314 describes a fault tolerant mirroring file server system in which one or more clients and two or more file servers are interconnected through a local area network and network protocol layers. The file servers include a control protocol with asymmetric responses such that a request for a file transfer from a client to a first file server is also received by and executed, or replicated, by the second file server and a request for a file transfer from a first file server to the client is normally executed by the first file server but is responded to by the second file server if the first file server does not respond.

It is apparent that the Kandasamy et al. '314 system will provide faster backup, mirroring and restoration than will the Rastogi et al. '449 simply because the Kandasamy et al. '314 system performs data transfers from the client to both servers in parallel while the Rastogi et al. '449 operates sequentially by writing to the primary server and having the primary server subsequently write a backup copy to the secondary server. It should also be noted that reads of data from one server to the client will occur at about the same speed in both systems because in both systems the backup server will wait to determine whether the primary server has responded or has failed before responding to a read request with the backup copy.

A review of the teachings of Kandasamy et al. '314, such as at column 3, lines 8-31; column 5, line 51 through column 6, line 19, and column 10, line 50 through column 11, line 10, shows that Kandasamy et al. '314 is similar to Rastogi et al. '449, and is thus distinguished from the present invention for the same reasons, in that the Kandasamy et al. '314 system captures, records and backs up data transfer requests only as self-contained entities. Stated another way, the Kandasamy et al. '314 system is in basic contrast from the present invention because the Kandasamy et al. '314 system does not and cannot capture and record the sub-operations

comprising the data transfer requests, but instead captures and records data transfer requests only as entities in themselves.

In this regard, it will be noted that the teachings of Kandasamy et al. '314 referred to above by column and line consistently refer to data transfer requests as self-contained entities and consistently describe each data transfer request as being comprised of a request and the associated data, which comprises a complete operation in itself. There is no suggestion in Kandasamy et al. '314 that an operation or transaction may be comprised of a sequence of data transfer requests or that a data transfer request may be comprised of a sequence of sub-operations.

Further in this regard, it must be noted that Kandasamy et al. '314 describes the clients and the first and second servers as being interconnected by and communicating through a local area network using a common and widely used multi-layer communications protocol. The communications protocol in turn imposes constraints on the form the communications can take and the rate at which the communications can occur. More specifically, the described communications protocol demands the use of communications "packets" wherein each packet is self-contained and complete entity in itself, including an address, and command or instruction that may pertain to or be the contents, and possibly a certain amount of data.

While this packet structure does not demand that each data transfer request be contained in a single packet, it does accommodate communications in which each data transfer request is a self-contained entity much more efficiently and readily than it does other forms of communication, such as streams of state engines wherein each state engine resides in a packet and the length of the stream is essentially unknown.

It must be further noted that the necessity to pass each communication through the multi-layer protocol at each end of the communication path, such as between a client and a data server, also imposes certain speed constraints on the Kandasamy et al. '314 system; that is, the system can transfer only so many "packets" per unit time regardless of the contents of the packets. This limitation therefore strongly biases the Kandasamy et al. '314 system to the method of operation in which each data transfer request is contained in as few packets as possible. This, in turn, strongly influences the Kandasamy et al. '314 system to the method of

operation wherein each data transfer request is a self-contained and complete operation rather than a sequence of sub-operations of unknown length.

For this reason, therefore, the communications method employed in the Kandasamy et al. '314 system reflects and strongly supports the conclusion that each data transfer request is a self-contained operation rather than a sequence of sub-operations, or states, or unknown length. As discussed above, this conclusion is strongly supported by the fact that Kandasamy et al. '314 consistently refers to data transfer requests as self-contained entities and consistently describes each data transfer request as being comprised of a request and the associated data, which comprises a complete operation in itself. There is no suggestion in Kandasamy et al. '314 that an operation or transaction may be comprised of a sequence of data transfer requests or that a data transfer request may be comprised of a sequence of sub-operations.

It is the belief and position of the Applicant that for the reasons discussed above the teachings of Kandasamy et al. '314 do not and cannot describe or suggest the present invention as recited in the claims under either or both of 35 U.S.C. § 102 or 35 U.S.C. § 103.

Continuing with Kandasamy et al. '314, and as discussed herein above, the Examiner appears to suggest that Kandasamy et al. '314 teaches a system similar to that of the present invention in that the first and second data servers comprise separate and independent but concurrently operating units that provide "instantaneous" responses to the clients or to a failure in one of the units. The Applicant disagrees for a number of reasons.

First, and as described above and in Kandasamy et al. '314, Kandasamy et al. '314 describes a fault tolerant mirroring file server system with asymmetric responses such that a request for a file transfer from a client to a first file server is concurrently received and executed by the second file server and a request for a file transfer from a first file server to the client is normally executed sequentially by the two servers, first by the first file server in direct response to the request and subsequently by the second file server if the first file server does not respond. While the Kandasamy et al. '314 does have a concurrent mode of operation between the two servers for write requests, the Kandasamy et al. '314 system is equally a sequential system as regards responses to read requests.

The system of the present invention is thereby fundamentally distinguished from the teachings of Kandasamy et al. '314 in that the control/processing sub-systems of the present invention operate concurrently and in parallel at all times rather than for only certain specific operations.

Further in this regard, it must be noted that the dual control/processing sub-systems of the present invention operate concurrently with one another, but independently of one another with each control/processing sub-system executing only the requests for transactions or operations that are directed to it, so that the full processing power of the two control/processing sub-systems is typically available at all times to process requests from clients. In contrast from the system of the present invention, the two data servers in the Kandasamy et al. '314 system both always respond to and operate upon the same data transfer requests. As a consequence, and while the two file servers of the Kandasamy et al. '314 system operate concurrently, they do not and cannot operate independently from one another. In fact, any attempt to cause the two data file servers of the Kandasamy et al. '314 system to operated independently from one another, such as on different data transfer requests at the same time, would be in direct contradiction to the fundamental principles of operation taught by Kandasamy et al. '314. Stated another way, the Kandasamy et al. '314 is operative as described and taught only when the two data file servers operate concurrently but in a first/second server relationship wherein both servers operate on the same data transfer request at the same time.

The system of the present invention is thereby fully and completely distinguished over and from the teachings of Kandasamy et al. '314 in that the control/processing sub-systems of the present invention operate concurrently and in parallel but independently from one another, with each operating separately on the data requests addressed to it, while the data servers of the Kandasamy et al. '314 system must operate together on each and every data transfer request.

Lastly, it must be noted that the first and second data servers of the Kandasamy et al. '314 system do not correspond either structurally or functionally with the state machine logging mechanism and state machine log mirroring mechanism of the present invention as the first and

second data servers of the Kandasamy et al. '314 system are identical but each is a separate, self-contained general purpose data file system. In contrast, the state machine logging mechanism and state machine log mirroring mechanism of the present invention are both dedicated purpose, specialized function mechanisms that are structurally and functionally different from one another and are directed to separate and distinctly different functions. In a like manner, and in complete contrast from the two identical and separate data file servers of the Kandasamy et al. '314 system, the state machine log mirroring mechanism of the present invention is functionally an integral element of the corresponding state machine logging mechanism, even though the state machine log mirroring system resides physically separate from the state machine logging mechanism as regards such support functions as the power supplies.

The system of the present invention is thereby still further fundamentally distinguished from the teachings of Kandasamy et al. '314 for the reasons just discussed.

In conclusion, therefore, it is the belief and position of the Applicant that for the reasons discussed above the teachings of Kandasamy et al. '314 do not and cannot describe or suggest the present invention as recited in the claims under either or both of 35 U.S.C. § 102 or 35 U.S.C. § 103. The Applicant therefore respectfully requests that the Examiner reconsider and withdraw all rejections of the claims over Kandasamy et al. '314 under either or both of 35 U.S.C. § 102 or 35 U.S.C. § 103.

Next considering the combination of Rastogi et al. '449 in further view of Kandasamy et al. '314, it is the belief and position of the Applicant that the combination of Rastogi et al. '449 with Kandasamy et al. '314 would not be apparent to those of ordinary skill in the relevant arts because of the very different fundamental nature and operation of the two systems. That is, the Rastogi et al. '449 system is a sequentially operating system wherein a data read or write transaction is first sent from the client to the primary server and is then subsequently backed up by transfer of a backup copy of the transaction from the primary server to the secondary server when the transaction has completed in the primary server. In fundamental contrast from Rastogi et al. '449, the Kandasamy et al. '314 is a parallel, concurrent system for data transfers from a client to the servers; that is, the data write request

and the data is transferred from the client and to both servers at the same time and both servers execute the data write request in parallel.

As such, the combination of Rastogi et al. '449 with Kandasamy et al. '314 would require the combination of two systems having completely and fundamentally different and conflicting modes of operation and would not result in any form of feasible system. As a consequence, the modification of Rastogi et al. '449's sequential system by the addition of the parallel operation taught by Kandasamy et al. '314 would require more than a "modification" to the Rastogi et al. '449 system and would require what would amount to a complete reconstruction of the Rastogi et al. '449 as a Kandasamy et al. '314 system. That is, and for example, the addition of the Kandasamy et al. '314 parallel method of operation to the Rastogi et al. '449 sequential system would result in nothing more than the Kandasamy et al. '314 system rather than in some form of hybrid sequential/parallel system.

Assuming solely for the purposes of discussion, and without any admission or agreement as regards the validity of such a combination, that some combination of Rastogi et al. '449 with Kandasamy et al. '314 were to be achieved, the result would be no more than a form of the Kandasamy et al. '314 system as discussed herein above. As such, the present invention as recited in the claims would be fully and patentably distinguished over and from the teachings of some combination of Rastogi et al. '449 with Kandasamy et al. '314 for the same reasons that the present invention is fully and patentably distinguished over and from the teachings of Kandasamy et al. '314.

It is, therefore, the belief and position of the Applicant that the combination of Rastogi et al. '449 in further view of Kandasamy et al. '314 is not a valid combination or teaching and, for the reasons discussed above, would not in any instance teach or even suggest the present invention as recited in the claims to those of ordinary skill in the arts under the requirements and provisions of 35 U.S.C. § 103.

The Applicant, therefore, respectfully requests that the Examiner reconsider and withdraw all rejections of the claims under either or both of 35 U.S.C. § 102 and 35 U.S.C. § 103 over the teachings of Rastogi et al. '449, the teachings of

Kandasamy et al. '314 and the teachings of Rastogi et al. '449 in further view of Kandasamy et al. '314, and the allowance of the claims.

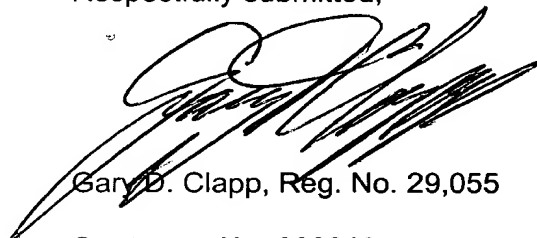
If any further amendment to this application is believed necessary to advance prosecution and place this case in allowable form, the Examiner is courteously solicited to contact the undersigned representative of the Applicant to discuss the same.

In view of the foregoing, it is respectfully submitted that the raised rejection(s) should be withdrawn and this application is now placed in a condition for allowance. Action to that end, in the form of an early Notice of Allowance, is courteously solicited by the Applicant at this time.

The Applicant respectfully requests that any outstanding objections or requirements, as to the form of this application, be held in abeyance until allowable subject matter is indicated for this case.

In the event that there are any fee deficiencies or additional fees are payable, please charge the same or credit any overpayment to our Deposit Account (Account No. 04-0213).

Respectfully submitted,



Gary D. Clapp, Reg. No. 29,055

Customer No. 020210
Davis & Bujold, P.L.L.C.
Fourth Floor
500 North Commercial Street
Manchester NH 03101-1151
Telephone 603-624-9220
Facsimile 603-624-9229
E-mail: patent@davisandbujold.com